JVM如何处理异常

异常包括两大组成要素，共同实现程序控制流的非正常转移：

* 抛出异常
* 捕获异常

## 抛出异常

### 显式抛出

显示抛出异常的主体是应用程序，指的是在程序用使用‘throw’关键字，手动将异常实例抛出

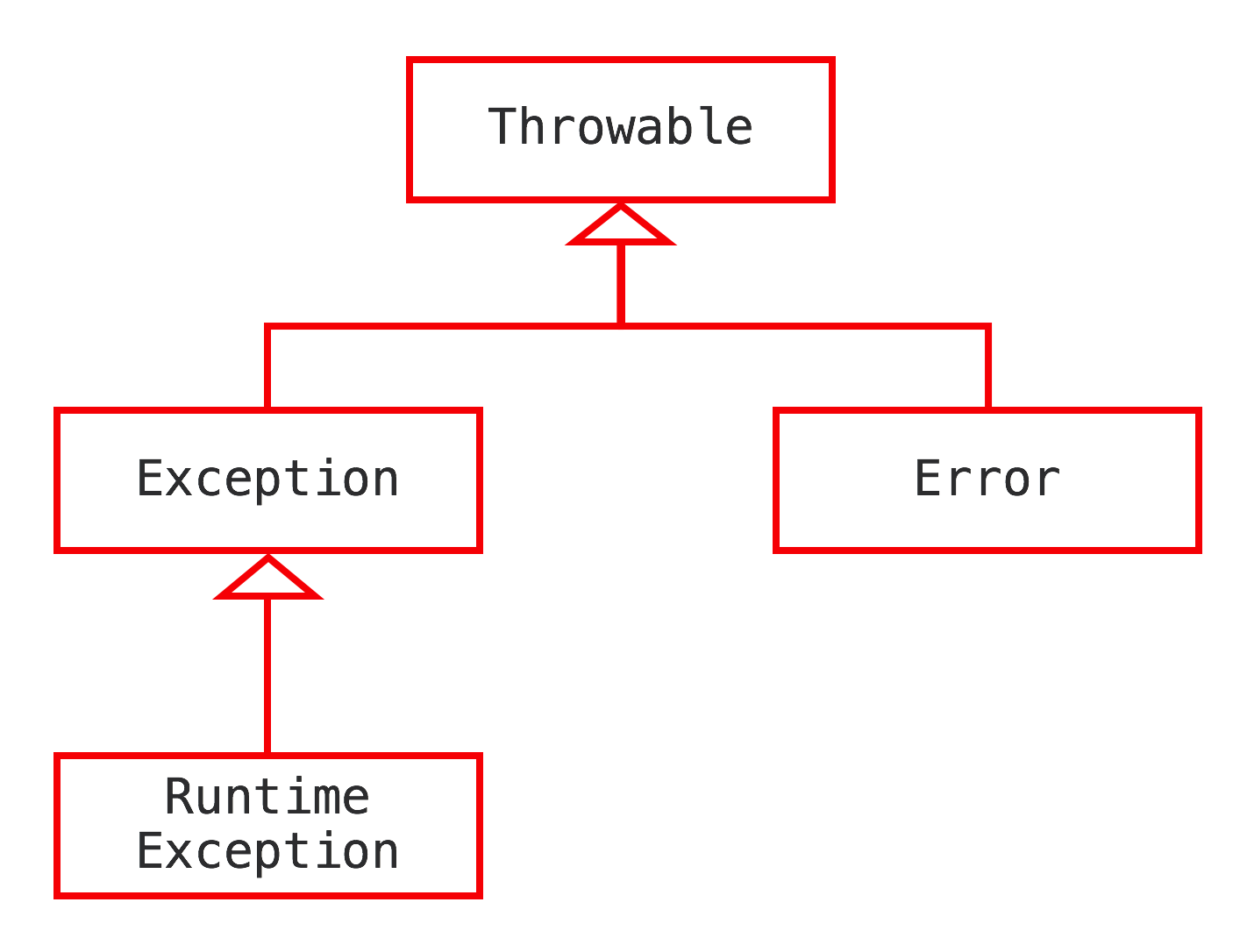
### 隐式抛出

Java虚拟机在执行的过程中，碰到无法继续执行的异常状态，自动抛出异常。

例如：java虚机在执行读取数据操作时，发现输入的索引是负数，抛出数组索引越界异常。

注意：在catch语句块中，java虚机会从上至下匹配异常处理器。因此，前面的catch代码块所捕获的异常不能覆盖后面的，否则编译器会报错。

如果try中遇到异常没有被捕获，那么finally代码会执行，然后再将异常向外抛出。



Error是程序不应捕获的异常，当程序处罚Error时候，它的执行状态已经无法恢复，需要终止线程甚至虚拟机。

Exception是程序需要捕获并且处理的异常。

-RuntimeException表示程序虽然无法运行，但是还可以抢救一下。

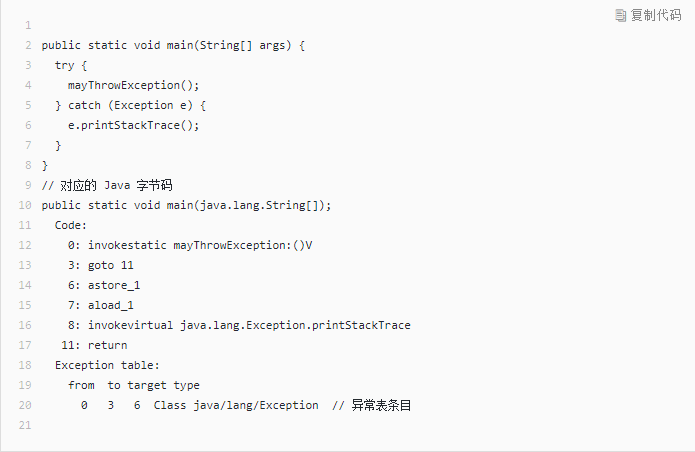
RuntimeException和Error属于java中的非检查异常（unchecked exception），其他异常则属于检查异常（checked exception）。

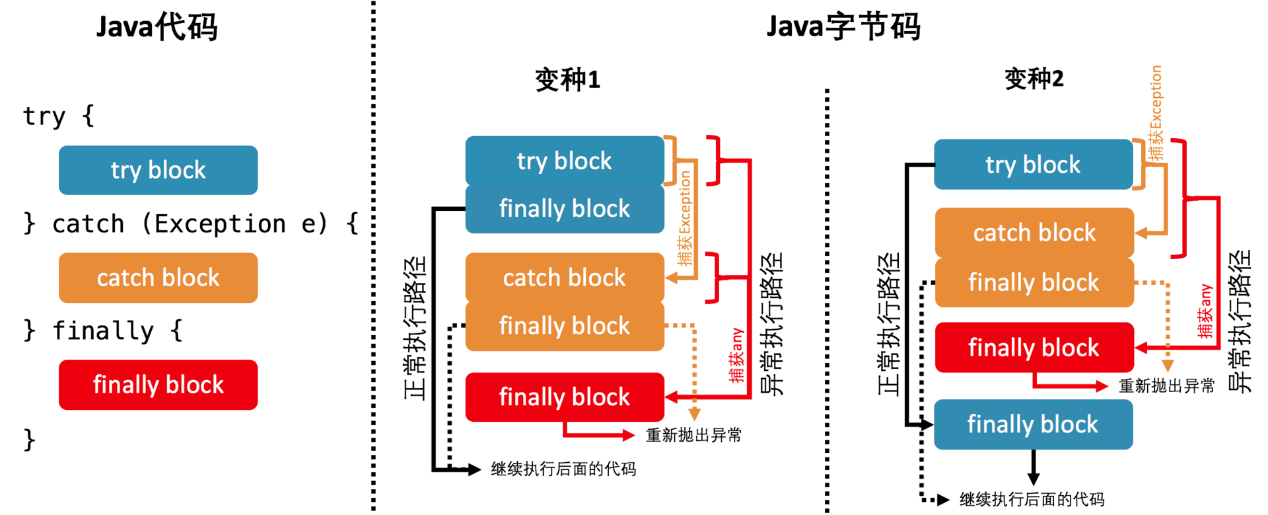
在java语法中，所有的检查异常都需要程序显示地去捕获。通常情况下，程序自定义的异常为检查异常，以便最大化的利用java编译时的编译时检查。

## 捕获异常

在编译生成的字节码中，每个方法都带了一张异常表。异常表中每一个条目代表一个异常处理器，并且由from指针、to指针、target指针以及所捕获的异常类型构成。这些指针的值为字节码索引，用以定位字节码。

From指针的t指针标识了该异常处理器所监控的范围。Target指针则指向异常处理器的起始位置。





public class cn.lxj.jvm40courses.course6.Foo {

public cn.lxj.jvm40courses.course6.Foo();

Code:

0: aload\_0

1: invokespecial #1 // Method java/lang/Object."<init>":()V

4: return

public void test();

Code:

0: aload\_0

1: iconst\_0

2: putfield #2 // Field tryBlock:I

5: aload\_0

6: iconst\_2

7: putfield #3 // Field finallyBlock:I

10: goto 35

13: astore\_1

14: aload\_0

15: iconst\_1

16: putfield #5 // Field catchBlock:I

19: aload\_0

20: iconst\_2

21: putfield #3 // Field finallyBlock:I

24: goto 35

27: astore\_2

28: aload\_0

29: iconst\_2

30: putfield #3 // Field finallyBlock:I

33: aload\_2

34: athrow

35: aload\_0

36: iconst\_3

37: putfield #6 // Field methodExit:I

40: return

Exception table:

from to target type

0 5 13 Class java/lang/Exception

0 5 27 any

13 19 27 any

}

可以看到编译结果包含三份finally代码块。其中，前两份分被位于try代码块和catch代码块的正常执行路径出口。最后一份则作为异常处理器，监控try代码块和catch代码块。它将捕获try代码块触发的、未被catch代码块捕获的异常、以及catch代码块触发的异常。

当catch代码块捕获了异常，那么finally捕获并且重抛的异常是后者（catch中新的异常）二将原来的try中的进入catch得 异常覆盖或者说是抛弃掉，这对代码调试来说很不利。

### Java7的Supressed异常以及语法糖

解决上面问题，catch语句块内的新异常。将一个异常附于另一个异常之上，因此，抛出的异常可以附带多个异常的信息。

因使用复杂，finally代码块缺少指向所捕获异常的引用，因此java提供了Try-with-resources语法糖——在字节码层面自动使用Supressed异常，

**public class Foo2 implements** AutoCloseable {  
 **private final String** name;  
  
 **public** Foo2(**String** name) {  
 **this**.name = name;  
 }  
  
 @Override  
 **public void** close() **throws Exception** {  
 **throw new** RuntimeException();  
 }  
  
 **public static void** main(**String**[] args) **throws Exception** {  
 **try** (**Foo2** foo1 = **new** Foo2("Foo1");  
 **Foo2** foo2 = **new** Foo2("Foo2");  
 **Foo2** foo3 = **new** Foo2("Foo3")) {  
 **throw new** RuntimeException("Initial");  
 }  
 }  
}

Exception in thread "main" java.lang.RuntimeException: Initial

at cn.lxj.jvm40courses.course6.Foo2.main(Foo2.java:24)

Suppressed: java.lang.RuntimeException

at cn.lxj.jvm40courses.course6.Foo2.close(Foo2.java:17)

at cn.lxj.jvm40courses.course6.Foo2.main(Foo2.java:25)

Suppressed: java.lang.RuntimeException

at cn.lxj.jvm40courses.course6.Foo2.close(Foo2.java:17)

at cn.lxj.jvm40courses.course6.Foo2.main(Foo2.java:25)

Suppressed: java.lang.RuntimeException

at cn.lxj.jvm40courses.course6.Foo2.close(Foo2.java:17)

at cn.lxj.jvm40courses.course6.Foo2.main(Foo2.java:25)